Caffe [1] uses Blobs [2] for storage and communicate data. It converts to and from Blob to other format in Data layers which often includes mean-subtraction, feature-scaling.

In addition, data often storages in efficient databases (LevelDB or LMDB), directly from memory and files from disk in HDF5 or another common file format.

To use Caffe for training data, we need to pre-process our data (mean-subtraction, feature-scaling, etc.) and put them in to proper file format like LMDB or HDF5. Still, we should clear on training data, validating data, testing data as well as the input data and labeled data.

# HDF5 File.

The generating HDF5 is very simple with Matlab code. However, HDF5 is not support parameter such as scaling. Thus, we need to pre-processing our data before put them into HDF5 format. Overall, HDF5 data layer requires a **.h5** file and a **.txt** file. The .h5 file contains your data and label, while the .txt file specifies the path(s) to the .h5 file(s). Following is an example of creating the .h5 file and its corresponding .txt file using python:

## Python

import h5py

import os

from \_\_future\_\_ import print\_function

DIR = "/PATH TO xxx.h5/"

h5\_fn = os.path.join(DIR, 'xxx.h5')

with h5py.File(h5\_fn, 'w') as f:

f['data'] = X

f['label1'] = Y1

f['label2'] = Y2

text\_fn = os.path.join(DIR, 'xxx.txt')

with open(text\_fn, 'w') as f:

print(h5\_fn, file = f)

Now you should have a .txt file and a .h5 file in your specified path.

The keys ‘data’, ‘label1’, ‘label2’ are keywords you defined for your data. You can have an arbitrary number of keywords, as long as you write the same keywords when you feed in your data to the hdf5 data layer. An example hdf5 data layer is like this:

layer {

name: "example"

type: "HDF5Data"

top: "data"

top: "label1"

top: "label2"

hdf5\_data\_param {

source: "/PATH TO .txt file/"

batch\_size: 100

}

}

Notice that the top blobs have the same keywords as when I created the .h5 file.

## Matlab.

Example for Matlab code for MNIST data.

|  |
| --- |
| clc  close all  clear all  %%  addpath mnistHelper;  addpath datasets;    % train-images.idx3-ubyte / train-labels.idx1-ubyte  images = loadMNISTImages('train-images-idx3-ubyte');  labels = loadMNISTLabels('train-labels-idx1-ubyte');    % reshape images to 4-D: [rows,col,channel,numbers]  trainData=reshape(images,[28 28 1 size(images,2)]);    % permute to [cols,rows,channel,numbers]  trainData=permute(trainData,[2 1 3 4]);    % permute lables to [labels, number of labels ]  trainLabels=permute(labels,[2,1]);    h5create('train.hdf5','/data',size(trainData),'Datatype','double');  h5create('train.hdf5','/label',size(trainLabels),'Datatype','double');    h5write('train.hdf5','/data',trainData);  h5write('train.hdf5','/label',trainLabels);    %%  % test images  images = loadMNISTImages('t10k-images-idx3-ubyte');  labels = loadMNISTLabels('t10k-labels-idx1-ubyte');    % reshape images to 4-D: [rows,col,channel,numbers]  testData=reshape(images,[28 28 1 size(images,2)]);    % permute to [cols,rows,channel,numbers]  testData=permute(testData,[2 1 3 4]);    % permute lables to [labels, number of labels ]  testLabels=permute(labels,[2,1]);    h5create('test.hdf5','/data',size(testData),'Datatype','double');  h5create('test.hdf5','/label',size(testLabels),'Datatype','double');    h5write('test.hdf5','/data',testData);  h5write('test.hdf5','/label',testLabels); |

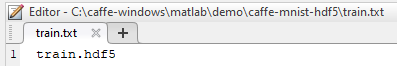
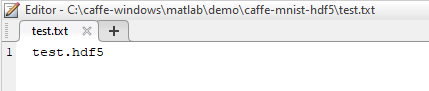
With two function of Load test and label data

|  |
| --- |
| function images = loadMNISTImages(filename)  %loadMNISTImages returns a 28x28x[number of MNIST images] matrix containing  %the raw MNIST images    fp = fopen(filename, 'rb');  assert(fp ~= -1, ['Could not open ', filename, '']);    magic = fread(fp, 1, 'int32', 0, 'ieee-be');  assert(magic == 2051, ['Bad magic number in ', filename, '']);    numImages = fread(fp, 1, 'int32', 0, 'ieee-be');  numRows = fread(fp, 1, 'int32', 0, 'ieee-be');  numCols = fread(fp, 1, 'int32', 0, 'ieee-be');    images = fread(fp, inf, 'unsigned char');  images = reshape(images, numCols, numRows, numImages);  images = permute(images,[2 1 3]);    fclose(fp);    % Reshape to #pixels x #examples  images = reshape(images, size(images, 1) \* size(images, 2), size(images, 3));  % Convert to double and rescale to [0,1]  images = double(images) / 255;    end |

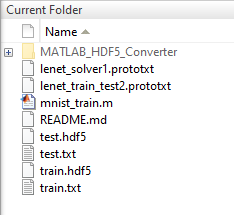
And

|  |
| --- |
| function labels = loadMNISTLabels(filename)  %loadMNISTLabels returns a [number of MNIST images]x1 matrix containing  %the labels for the MNIST images    fp = fopen(filename, 'rb');  assert(fp ~= -1, ['Could not open ', filename, '']);    magic = fread(fp, 1, 'int32', 0, 'ieee-be');  assert(magic == 2049, ['Bad magic number in ', filename, '']);    numLabels = fread(fp, 1, 'int32', 0, 'ieee-be');    labels = fread(fp, inf, 'unsigned char');    assert(size(labels,1) == numLabels, 'Mismatch in label count');    fclose(fp);    end |

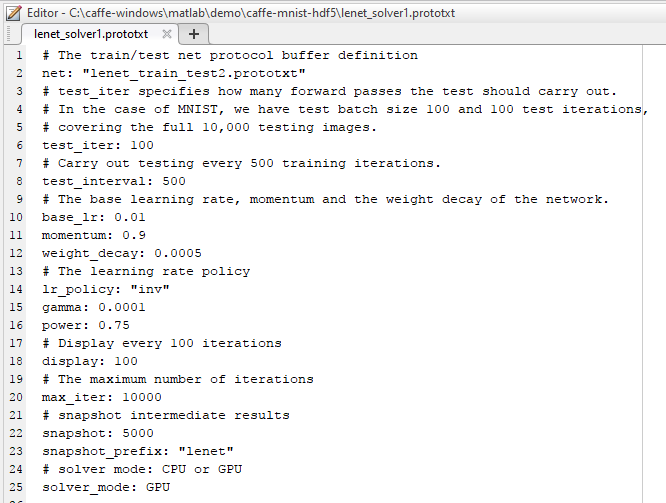
It should be noted that, Caffe cannot read the \*.hdf5 file directly. Instead, we need to make two .txt files to specify the location of the HDF5 in its first row.

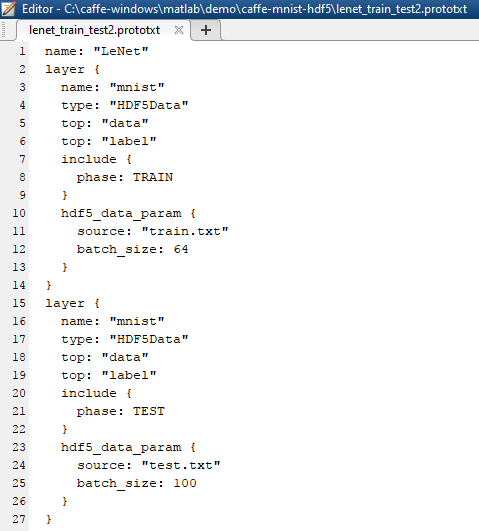


The location of file is given as



And the corresponding two prototxt files are





[1] Caffe Data: <http://caffe.berkeleyvision.org/tutorial/data.html>

[2] http://caffe.berkeleyvision.org/tutorial/net\_layer\_blob.html#blob-storage-and-communication

[3] Python: https://ceciliavision.wordpress.com/2016/03/21/caffe-hdf5-layer/